Uvod u Python grafičke okvire i Tkinter

Kada je reč o grafičkim radnim okvirima (GUI frameworks), za Python postoji veći izbor rešenja. Od svih njih, Tkinter je jedini ugrađen i dolazi sa instalacijom Pythona. Neki od tih radnih okvira se mogu koristiti širom drugih platformi, dok su neki namenski razvijeni za određene platforme. Detaljnija lista radnih okvira i platformi na kojima se oni mogu koristiti može se naći na ovom linku. Neki od GUI radnih okvira koje ćemo obraditi u ovoj nastavnoj jedinici su: Kivy, PyQT, WxPython i Tkinter. Kroz primere ćemo obraditi početni Hello World primer za svaki od ovih radnih okvira, a u ostatku kursa fokusiraćemo se na Tkinter.

Kivy

Kivy je Python modul koji omogućava kreiranje aplikacija za različite platforme koristeći upravo programski jezik Python. Omogućava vrlo lako korišćenje istog koda kako bismo napravili aplikacije za Windows, Linux, iOS, Android ili bilo koji drugi poznatiji operativni sistem. Upravo je takva filozofija – *jedan kod koji radi na svim platformama* – bila pokretač za razvoj ovog GUI radnog okvira. Ako je reč o sistemima koji funkcionišu na uređajima osetljivim na dodir – podržava prepoznavanje više dodira po ekranu odjednom. Njegov izgled se malo razlikuje od nativnog izgleda kontrola na platformi. Besplatan je za korišćenje i otvorenog je koda. Zvanična dokumentacija ovog modula dostupna je preko ovog linka.

Za instalaciju Kivyja koristimo pip alat, izvršavajući sledeće dve linije (jednu za drugom). U ovom pozivu pip komande prvi put možemo videti instalaciju više paketa odjednom, odnosno u jednom pozivu pip komande. U ovom slučaju imena paketa odvajamo praznim mestom:

```
pip install docutils pygments pypiwin32 kivy.deps.sdl2 kivy.deps.glew
pip install kivy
```

Prva linija se odnosi na instaliranje modula od kojih sam Kivy zavisi, dok se druga linija odnosi na instalaciju samog modula.

Za pokretanje Hello World primera, kreirajmo prvo prazan fajl po imenu kivy_test.py. Najčešći način za kreiranje aplikacija u Kivyju, a i generalno u drugim grafičkim radnim okvirima, jeste koristeći OOP koncept. Pa tako, kreiramo klasu koja će predstavljati našu aplikaciju. Svaka naša Kivy aplikacija mora nasleđivati ugrađenu App klasu iz modula kivy.app. Pa tako, sledi sledeći set import naredbi:

```
import kivy
from kivy.app import App
from kivy.uix.label import Label
```

Pored App klase uvezli smo i kontrolu (vidžet) – Label, odnosno labelu u kojoj ćemo ispisati *Hello World* tekst.

U klasi koja će predstavljati našu aplikaciju moramo uvek definisati i build() funkciju u kojoj se definiše koje će se to tačno kontrole pojaviti na ekranu.

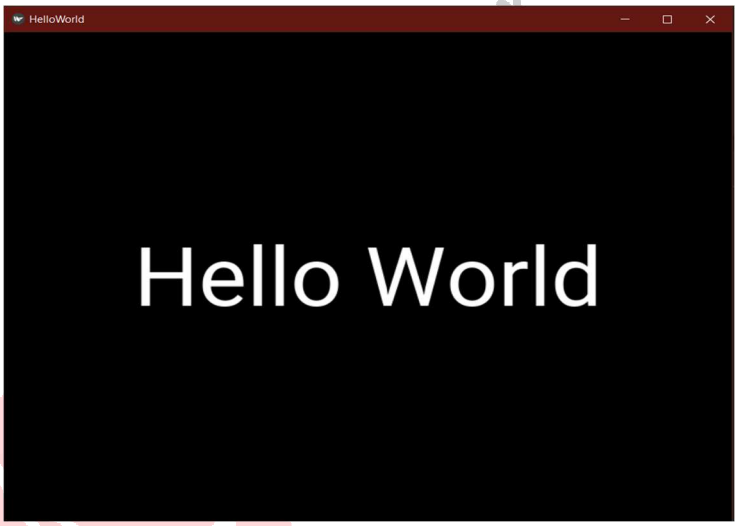
kivy_test.py fajl

```
import kivy
from kivy.app import App
from kivy.uix.label import Label

class HelloWorld(App):
    def build(self):
        return Label(text="Hello World", font_size = 100)

if __name__ == "__main__":
    HelloWorld().run()
```

U ovom primeru vidimo da u funkciji build() imamo povratnu vrednost. Povratna vrednost funkcije build() je element koji zapravo i želimo da prikažemo na ekranu. U ovom slučaju je to samo labela, ali može biti i bilo koji druga kontrola – ili, što je češće slučaj – posebna layout kontrola koji predstavlja kontejner (skup) kontrola koje želimo da prikažemo. Nakon definisanja klase HelloWorld() ostaje još da je pokrenemo naredbom HelloWorld().run(). Ako smo uspešno instalirali Kivy, dobićemo ovakav ispis na ekranu:



Slika 2.1. Prikaz aplikacije kreirane u Kivy-ju

Podrazumevano ime prozora naše aplikacije se nasleđuje od imena klase, kao što možemo i videti sa slike.

Napomena

Na našim slikama primera se može primetiti da je boja samog prozora crvena. Ovo ne treba da zbuni polaznika ako se razlikuje od boje prozora aplikacija koje on pokreće na svom računaru, jer je reč o temi Windows 10, koju je autor kursa postavio na crveno. Podrazumevana tema prilikom instalacije Windows 10 operativnog sistema je bela (dakle, boja prozora aplikacija je bela).

PyQt

PyQt predstavlja vezu između Pythona i Qt radnog okvira. Qt je vrlo popularan i moćan grafički radni okvir napisan u programskom jeziku C++; podržava razvoj grafičkih aplikacija na više različitih platformi. PyQt je razvijen je razvila kompanija RiverBank Computing Ltd; postoje dve verzije:

- PyQt4 – verzija koja je prvenstveno bila bazirana na Qt 4.x, a kasnije je dodat deo API-a iz Qt5.x;
- PyQt5 – verzija koja je bazirana na verziji Qt 5.x.

U našem primeru ćemo koristiti PyQt5, koji je noviji i predstavlja budućnost razvoja ove biblioteke.

Prednost Qt radnog okvira, pa samim tim i PyQt-ja, ogleda se u velikom broju već ugrađenih modula spremnih za korišćenje. Takođe je kompatibilan sa svim poznatijim platformama (Windows, Linux, iOS, Android...). Podrazumevani vizuelni stil ovih aplikacija je zapravo i nativni izgled aplikacija na platformi gde se aplikacija i razvija.

Bitna stvar kod razvoja aplikacija koristeći PyQt/Qt način jeste: ako planiramo komercijalni razvoj aplikacije u PyQt radnom okviru, moramo nabaviti odgovarajuću licencu. Najveći konkurent ovoj biblioteci je Qt for Python – biblioteka razvijena upravo od Qt-ja, koja kao takva predstavlja zvanični Python modul za Qt. Bitna razlika između ove dve biblioteke je što Qt for Python možemo besplatno koristiti i u komercijalne svrhe. Ali kako je Qt for Python još uvek u ranom razvoju, za naš Hello World primer koristićemo PyQt5. Zvanična dokumentacija kompanije RiverBank za PyQt5 se nalazi na ovom linku.

Pre kreiranja aplikacije moramo instalirati PyQt5 pip komandom:

```
pip install pyqt5
```

Nakon toga, kreirajmo fajl PyQt5_test.py, u kojem ćemo prvenstveno uvesti neophodne klase za rad:

```
from PyQt5.QtWidgets import QApplication, QMainWindow, QLabel
```

Kada god kreiramo aplikaciju pomoću PyQt bilbioteke, moramo raditi na objektu klase QApplication – dakle, svaka naša aplikacija je zapravo instanca ove klase. QMainWindow klasa se odnosi na prozor te aplikacije koji će sadržati QLabel objekat, odnosno labelu. Kod za Hello World aplikaciju je sledeći:

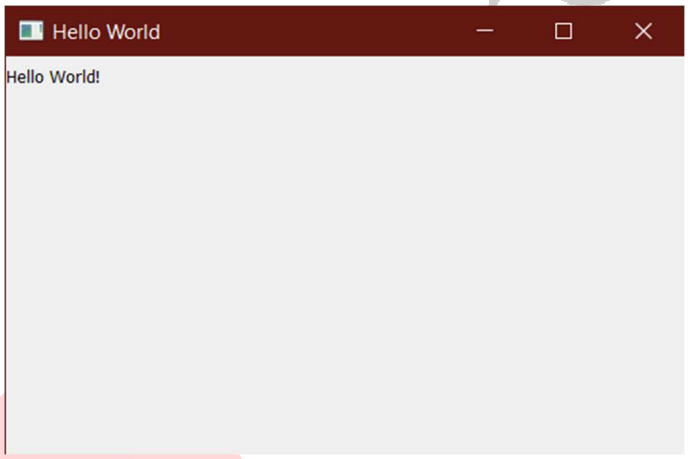
PyQt5_test.py fajl

```
from PyQt5.QtWidgets import QApplication, QMainWindow, QLabel

app = QApplication([])
win = QMainWindow()
win.setGeometry(400,400,500,300)
win.setWindowTitle("Hello World")
label = QLabel('Hello World!', parent = win)
win.show()
app.exec_()
```

Konstruktoru klase QApplication() smo prosledili praznu listu kao argument. Argumenti ovom konstruktoru su argumenti koje prosleđujemo Python skripti koji se nalaze u naredbi sys.argv, pa tako, instanciranje PyQt5 aplikacijskog objekta može izgledati ovako: app = QApplication(sys.argv). Naravno, pre korišćenja ovakvog načina potrebno je uvesti ugrađenu biblioteku sys. Pošto smo inicijalizovali aplikaciju, moramo kreirati i prozor naredbom win = QMainWindow(). Ovaj prozor će sadržati labelu sa tekstom Hello World. Tom prozoru ćemo proizvoljno odrediti dimenzije metodom setGeometry(), kojoj prosleđujemo x i y koordinate u odnosu gornju levu ivicu ekrana i željenu širinu i visinu prozora, dok ćemo naziv tom prozoru postaviti metodom setWindowTitle(). Pošto imamo prozor, definisaćemo labelu naredbom label = QLabel('Hello World!', parent = win), gde konstruktoru QLabel klase prosleđujemo promenljivu win, odnosno, definišemo kojoj će se kontroli labela pridružiti. Pored tog argumenta, prosleđujemo i sam tekst labele.

Ako bismo u ovom trenutku pokrenuli naš PyQt5_test.py fajl, ne bismo ništa dobili, jer još uvek nismo prikazali naš prozor. To činimo metodom show() nad QMainWindow objektom. Ako bismo sada pokrenuli fajl, videli bismo da se prozor tek na trenutak pojavljuje i odmah gasi. Da bismo ga zadržali na ekranu, potrebno je pozvati exec_() metodu aplikacijskog objekta koji našu aplikaciju stavlja u beskonačnu petlju i time zadržava prozor na ekranu dokle god ga korisnik ne isključi. Grafički prikaz naše aplikacije izgleda ovako:



Slika 2.2. Prikaz aplikacije kreirane u PyQt5 okviru

WxPython

WxPython grafički okvir je otvorenog koda sa mogućnošću korišćenja na različitim platformama kao što su Windows, macOS i Unix. WxPython je baziran na popularnoj biblioteci WxWidgets (nekadašnji WxWindows) napisanoj u programskom jeziku C++. Moguće je besplatno korišćenje wxPython biblioteke u komercijalne svrhe.

Zvanična dokumentacija se nalazi na ovom linku.

Pre korišćenja wxPythona, moramo ga instalirati komandom: pip install wxpython.

Za kreiranje Hello World aplikacije, u fajlu wx_test.py moramo uvesti modul import wx. Kao što smo videli i u prethodnim grafičkim okvirima, i u wxPythonu moramo prvo kreirati objekat aplikacije linijom app = wx.App(). Opet, kao i u prethodnim primerima, da bi se Hello World tekst prikazao na ekranu, moramo kreirati prozor ili drugi kontejnerski objekat koji će sadržati

kontrolu sa tekstom. U slučaju wxPythona, to je window = wx.Frame(None, title = "Hello World Window", size = (300,200)) objekat. Prvi argument, None, govori o tome kako wx.Frame objekat nema nijednu kontrolu koja je hijerarhijski viša od nje (parent). Ostali argumenti se odnose na tekst frejma, odnosno prozora, kao i na njegove dimenzije. Za labele u wxPythonu koristimo manje intuitivnu klasu po imenu wx.StaticText, koja se odnosi na rad sa labelama, i to na sledeći način:

```
label = wx.StaticText(window, label = "Hello World", pos = (100,50))
```

Prvo što konstruktoru klase wx.StaticText() prosleđujemo je prozor na kojem će se kontrola prikazati. Dalje nagoveštavamo da je reč o labeli ključnim argumentom label i dodeljujemo koordinate gde će kontrola prikazati na prozoru.

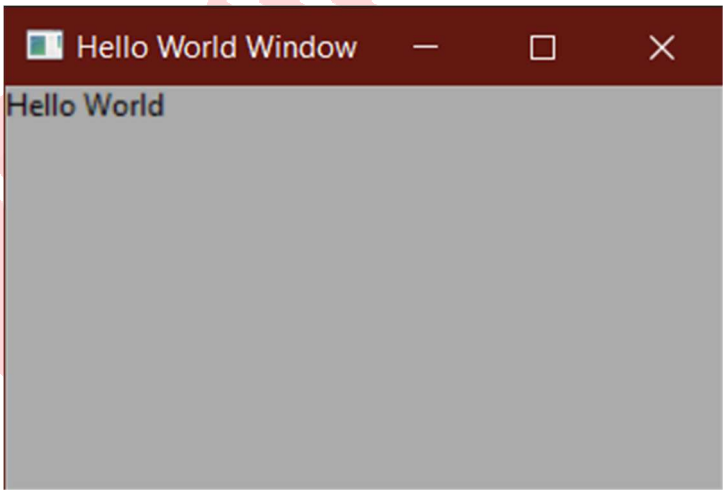
Slično kao i u slučaju PyQt radnog okvira, naš prozor moramo pokazati metodom Show(), a čitavu wx.App() aplikaciju smestiti u beskonačnu petlju naredbom app.MainLoop(), kako se aplikacija ne bi ugasila odmah po pokretanju fajla.

wx_test.py fajl

```
import wx

app = wx.App()
window = wx.Frame(None, title = "Hello World Window", size = (300,200))
label = wx.StaticText(window, label = "Hello World", pos = (100,50))
window.Show()
app.MainLoop()
```

Grafički prikaz wxPython aplikacije izgleda ovako:



Slika 2.3. Prikaz aplikacije kreirane u wxPython-u

Izmeniti kod tako što ćete pomoću Wx modula napraviti okvir sa naslovom Big Window veličine 500x500 udaljen 500 piksela od leve ivice i 500 od gornje ivice ekrana.u kome treba da se nalazi labela sa tekstom Mali tekst.

Pitanje

Ispravan način za inicijalizovanje wx aplikacije je:

- wx.Application()
- wxPython.App()
- **wx.App()**

Objašnjenje:

Tačan odgovor je da se wx aplikacija instancira wx.App() metodom.

Tkinter

Tkinter je ugrađeni Python modul za rad sa grafičkim interfejsima. Pošto se Python može koristiti na više platformi – isto važi i za Tkinter aplikacije. Samo ime je nastalo iz *Tk interface*. Tk GUI interfejs je originalno razvijen za Unix kao dodatak za TCL programski jezik (tool command language); popularnost među programerima je stekao tokom devedesetih godina prošlog veka. Deo je standardnih Python biblioteka još od Python verzije 1.1 i kao takav dopušta besplatno kreiranje komercijalnih aplikacija. Zvanična Tkinter dokumentacija se može naći na ovom linku.

Za pokretanje Hello World primera u Tkinteru, kreirajmo prvo fajl po imenu tkinter_test.py sa sledećim kodom:

tkinter_test.py fajl

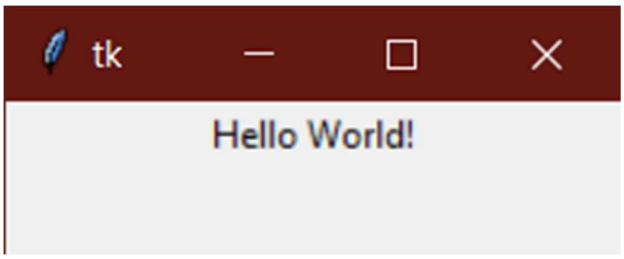
```
import tkinter as tk
root_window = tk.Tk()
root_window.minsize(200, 50)
label = tk.Label(root_window, text="Hello World!")
label.pack()
root_window.mainloop()
```

Najpre za rad sa Tkinterom uvozimo istoimenu biblioteku, kojoj dodajemo tk kao alijas radi lakšeg korišćenja.

Kao i u prethodnim primerima, potrebno je prvo kreirati neku vrstu aplikacijskog objekta. U slučaju Tkintera, to se vrši linijom: root_window = tk.Tk(). Takođe, u slučaju Tkintera, tk.Tk() naredba nam označava i glavni prozor naše aplikacije. Metodom minsize() jednostavno preciziramo minimalne dimenzije prozora naše aplikacije prilikom pokretanja. Za prikaz labele koristimo tk.Label() klasu kojoj prosleđujemo naš root_window objekat kako bismo upravo na njemu i prikazali našu labelu i pored toga prosleđujemo tekst labele. Ovaj root_window objekat smo mogli proslediti i koristeći master ključni argument.

Ono što je bitno napomenuti je da nakon te linije – labela još uvek nije prikazana na ekranu. Ovo ćemo izvršiti metodom pack(), kojom upravo to i činimo. Ova metoda, pored toga što prikazuje kontrolu nad kojom je pozvana, upravlja organizacijom kontrola u prozoru. Jedna je od tri metode (pack, grid, place) upravljača grafičkog prostora (geometry manager) u Tkinteru.

Kao i u prethodnim primerima, moramo metodom root_window.mainloop() svoju aplikaciju smestiti u beskonačnu petlju kako bismo uopšte videli svoj prozor. Prozor naše Hello World Tkinter grafičke aplikacije izgleda ovako:



Slika 2.4 Prikaz Hello World aplikacije kreirane u Tkinteru

Izmeniti kod tako što ćete pomoću Tkinter modula napraviti labelu u koju ćete upisati Dobar dan! I odrediti da je minimalna dimanzija prozora 300x200.

Neki od ostalih grafičkih okvira koji su takođe popularni a nisu obrađeni u ovoj nastavnoj jedinici su PyGUI i PySide.

Rezime

- Kada je reč o grafičkim radnim okvirima (GUI frameworks), za Python postoji veći izbor rešenja. Od svih njih, Tkinter je jedini ugrađen i dolazi sa instalacijom Pythona.
- Kivy je Python modul koji omogućava kreiranje aplikacija za različite platforme koristeći upravo programski jezik Python. Omogućava vrlo lako korišćenje istog koda kako bismo napravili aplikacije za Windows, Linux, iOS, Android ili bilo koji drugi poznatiji operativni sistem.
- PyQt predstavlja vezu između Pythona i Qt radnog okvira. Qt je vrlo popularan i moćan grafički radni okvir pisan u programskom jeziku C++; podržava razvoj grafičkih aplikacija na više različitih platformi.
- WxPython grafički okvir je otvorenog koda sa mogućnošću korišćenja na različitim platformama kao što su Windows, macOS i Unix. WxPython je baziran na popularnoj biblioteci WxWidgets (nekadašnji WxWindows) napisanoj u programskom jeziku C++.
- Tkinter je ugrađeni Python modul za rad sa grafičkim interfejsima. Pošto se Python može koristiti na više platformi – isto važi i za Tkinter aplikacije. Samo ime je nastalo iz *Tk interface*.